Visual Comput (2006) 22: 332–345
DOI 10.1007/s00371-006-0009-3

ORIGINAL ARTICLE

Hyung Seok Kim
Chris Joslin
Thomas Di Giacomo
Stephane Garchery
Nadia Magnenat-Thalmann

Device-based decision-making for adaptation of three-dimensional content

Published online: 20 April 2006

H.S. Kim*(✉) · C. Joslin** ·
T. Di Giacomo · S. Garchery ·
N. Magnenat-Thalmann
MIRALab – University of Geneva,
Switzerland
hyung.kim@acm.org;
{Giacomo, Garchery,
Thalmann}@miralab.unige.ch;
chris_joslin@carleton.ca

Abstract The goal of this research was the creation of an adaptation mechanism for the delivery of three-dimensional content. The adaptation of content, for various network and terminal capabilities – as well as for different user preferences, is a key feature that needs to be investigated. Current state-of-the art research of the adaptation shows promising results for specific tasks and limited types of content, but is still not well-suited for massive heterogeneous environments. In this research, we present a method for transmitting adapted three-dimensional content to multiple target devices. This paper presents some theoretical and practical methods for adapting three-dimensional content, which includes shapes and animation. We also discuss practical details of the integration of our methods into MPEG-21 and MPEG-4 architectures.

Keywords Multi-resolution model · Virtual human animation · Content adaption · Benchmarking · MPEG-21

1 Introduction

1.1 Targeting a context

Three-dimensional (3D) representation is one of the cornerstones of computer graphics and multi-media content. Advances in this domain, coupled with the highly fuelled progression of 3D graphics cards has pushed the complexity of these representations into a whole new arena whereby a single real-time model can consist of more than a million polygons. Huge architectural buildings, everyday objects, even human beings themselves can be represented using 3D graphics in such detail that it is difficult to distinguish between real and virtual images.

Concurrently, and much towards the other end of the scale, many devices (personal digital assistants (PDAs), mobile phones, laptops, etc.) are now "3D-enabled"; capable of enhancing a user's experience and providing much more depth to the presented information. In many cases, these devices access the same content from the same service provider (for example: providing virtual maps/guides, multi-user games, etc.) and it is this broadness of content and the heterogeneity of devices (in terms of performance, capability, network connection, etc.) that is the main concern in a continuously expanding market. It is also the concern of the user to obtain the best quality for their device, i.e., a general expectation of any device of higher performance is that overall the quality of the experience will be better.

1.2 Media adaptation

Media adaptation offers a solution to these current industry woes and whilst most of the more common examples are in the area of video and audio, the popularization of games and virtual environments shows that there is also an emerging arena for 3D graphics. Whilst there are many considerations for 3D graphics as a whole, being a very broad domain, we focus our attention in this paper mainly on the adaptation of graphical representation (with an

* Presently at Konkuk University, Korea
** Presently at Carleton University, Canada

emphasis on virtual humans), and introducing animation adaptation. In addition, whilst the research area of multi-resolution models [22] is one that has been significantly explored, we introduce it into the domain of MPEG-21 digital item adaptation in order to consider the constraints imposed by multiple targets, and the adaptation from a single media item.

Our main research goal was to devise a method for multi-resolution representation/adaptation to transfer and render a virtual human (face and body) on heterogeneous target devices with the following considerations:

- **Range of complexity** – As the model will not be targeted specifically for a particular network or device, we needed to define a generic method. In essence, this means the ability to represent a virtual human from its lowest possible representation towards its most complex form; this is in opposition to creating a range of models and selecting the most appropriate one.
- **Device capability** – We defined a generic method for assessing the capability of the device to enable the closest approximation. As the target device is not specified, it should be assumed that this assessment should also be heterogeneous.
- **Network capacity** – Based on an assessment of the network condition, we used a generic method for adaptation that allows for both a specific bandwidth and maintains a certain level of quality of service.
- **Interoperability** – It is obviously necessary that whilst the server needs to be aware of an adaptation of the content towards a specific device (as it is performing the task), client awareness should not be required. Hence, regardless of whether content is adapted or not, it should be usable on the target device without any specific indicators. The reasons are many, but the main one is that the decoding process should not require additional information as this would create (a) a huge dependency and (b) updates to an already defined schema; in addition it is also not necessary as long as the process is thoroughly considered.
- **Animatable models** – The models themselves are not just representations, they must be also animated, which means that we needed to consider ramifications at the joint (body) and control point level (face) of simplification.
- **Compact data form** – Adaptation for a specific terminal should not severely impact network performance and vice versa (e.g. adapting towards a specific terminal should not increase the data transferred on the network). In addition, the data should be represented in compressed form wherever possible.

1.3 MPEG-21 digital item adaptation

The Motion Picture Experts Group's (MPEG), Section 21, Part 7 on digital item adaptation (DIA) (collectively known as MPEG-21 DIA) provides a tidy solution to the aforementioned constraints. The term digital item was coined to cover basically all media types, and include essentially the media (video, audio, 2D/3D graphics, descriptions, etc.) and other associated items (e.g. licenses, play lists, etc.). Adaptation can be performed on any or all of these digital items depending on the method employed. The standard [32] specifically focusing on DIA contains references to methods for adaptation (described in Sect. 3) and context; although both areas are highly generic, enabling implementation of the standard without restriction to specific practices. Context, in this case, is used to define how the resulting digital should end up on the user's terminal and covers an entire range of definitions. A brief summary of these definitions is given below.

- **Network** – contains the definitions specifying the network, including maximum capacity, and current usage.
- **Terminal** – this defines the user's terminal itself; definitions range from the resolution of the terminal's screen to its storage capacity.
- **User constraints** – these define the user's personal impairments, for example if they are visually impaired, this can be taken into account in the adaptation of content.
- **User preferences** – used to define the user's actual preferences.
- **Mobility** – used to define the terminal's actual position, the location, possibly the local language spoken

Some or all of the contextual information is sent to the server in order to facilitate the adaptation process. The interpretation of these contextual elements is not standardized, only their definition. In this paper, we touch upon only a few of these elements, specifically the definition of terminal capability, and network capacity. Here we focus on how these contextual elements relate to the adaptation of graphical elements (both representation and animation) with an emphasis on virtual humans. We focus on both adaptations for facial and body animation, extending current state-of-the-art work by taking into account the geometry and a more general context, which includes terminal capabilities. In Sect. 2, we present related work on mesh and animation adaptation (with only simple context). Sect. 3 discusses the issues involving heterogeneous targets and Sect. 4 presents the experimental results with MPEG schemes. Section 5 concludes the paper.

2 Related work

2.1 Geometry representation

The real-time rendering of complex objects and multi-resolution approaches have been widely investigated and applied. Starting from mesh simplification approaches applied for the complex mesh obtained from the laser scanner [12, 27], mesh simplification methods are applied

to real-time rendering by reducing rendering complexity while preserving as much visual detail as possible. For the simplification process, the main issues have been focused on identifying and measuring features. Traditionally, object level features have been investigated, which include vertex distance [25], curvatures [35], geometric measurement [9], texture distance [10], and distance in image space [31].

After the simplification, the model is represented in a hierarchical way to be processed in real-time [22]. A discrete multi-resolution structure has been broadly adopted in application due to its simplicity and efficiency. In addition to the discrete structure, there have also been other hierarchical structures including ones which based on parametric surface representations [16] and vertex-tree based representations [26].

Previous structures have focused on issues of the runtime adaptation of geometry. Although most of methods have strong points in efficiently managing complexities in real-time, some improvement is required for use in heterogeneous networked environments. There have been a few approaches to adapt multi-resolution techniques for the transmission of complex shapes including progressive meshes (PM) [8, 17, 25]. Those approaches provide relatively good results but need relatively high cost reconstruction on the client side, which places some limitation on using low-end devices such as PDA's or mobile phones. Also, when the issue comes to the binary encoded stream, some methods require more investigation to make it possible to adapt the contents in the binary state while preserving the rendering quality.

2.2 Animation representation

In a similar manner to geometry simplification, motions can be simplified in a visually lossless way under certain circumstances and, therefore, methods to control the trade-off of animation realism versus processing and storage requirements are being investigated by researchers.

In motion control, with regards to the selection of motion complexity over realism, it is of paramount importance to overview the basics of human visual perception, especially human motion perception, in the case of animation. Adelson [1] discusses some major issues such as apparent motions, spatial-temporal receptive fields, and short and long range perception mechanisms. Distler et al. [15] study the estimations of speed that the brain is providing using specific perceptual cues, and report that both temporal-frequency and distance play a significant role in generating velocity consistency and estimation. Directly related to skeleton and human-like motions, experiments by Kourtzi et al. [30] suggest that human movement perception is easily recognized and based on intrinsic biomechanical constraints.

In computer animation, as mentioned by Berka [6], the accuracy of motions that are "too" fast, "too" far away, or "too" numerous in a virtual scene can be reduced without impacting their efficiency. Hence, because of its processing requirements, physically-based animation can highly gain from LoD animation systems, for instance as proposed by Carlson et al. [7] with a combination of three different animation systems, ranging from pure kinematics to dynamics, or methods proposed by Hutchinson et al. [28] that adopt a rough mass-spring network while locally refining it when the spring-angles constraints are enforced. With the same general idea, Debunne et al. [11] adapt a finite-element mesh where forces are applied for higher control on the deformations in relevant regions. Regarding non-physically-based animation methods, the inherent hierarchies of articulated bodies are very appropriate for animation control using LoD methods. Di Giacomo et al. [13] apply two animation methods with levels of depth in the hierarchy of animated branches of trees, while for human-like structures, Granieri et al. [21] investigate LoD for animation by decreasing the sampling frequency of motions and the number of degrees of freedoms per joints, i.e. the number of dimensions in which they can move. In more recent work, Giang et al. [20] discuss integration of LoD with LoA, and Joslin et al. use visual experiments to cluster animations for the actual observer to animation distances. Moreover, Di Giacomo et al. [14] propose to control the number of joints with a level of articulations and regions of interest, while Ahn et al. [3] preprocess joint postures in clusters to limit computations due to transformations at run-time.

3 Adaptation overview

3.1 Introduction

Adaptation within the MPEG framework is a relatively new topic and as such only a few generic methods have been proposed [5], mainly dealing with audio/video media; whilst 3D graphics has received little attention. Here we present our research on the development of an adaptation schema using both MPEG-4 and MPEG-21 schemas. We focus on the global aspect of device capability in networked environments. As a measurement of device capability, we place emphasis on utilizing a benchmarking process and demonstrate a scheme now standardized in MPEG 21 DIA.

3.2 Overall configuration

Whilst MPEG-21 DIA [32] is not only restricted to a client/server architecture, it is currently the most commonly used for most applications; however some peer-to-peer applications are emerging.

Even client-side only adaptations, whereby stored data is directly adapted according to a specific context, are also being explored on the basis of providing the more

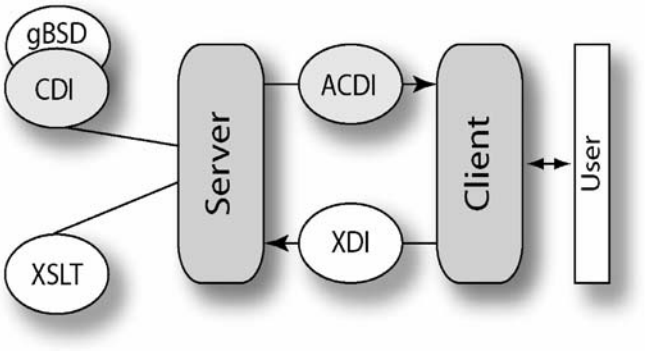

Fig. 1. Basic architecture overview

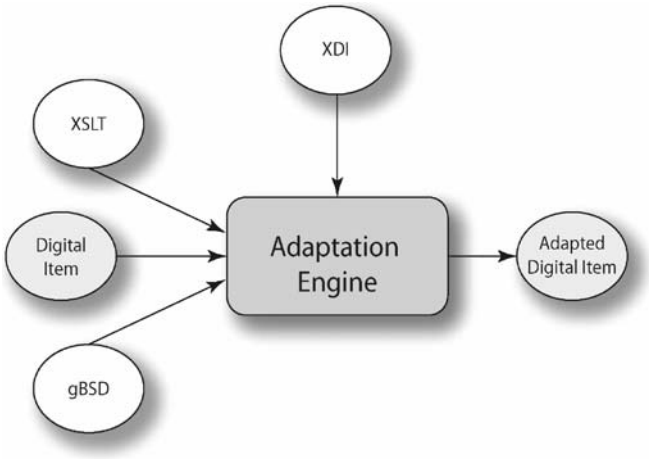

Fig. 2. Overall adaptation process

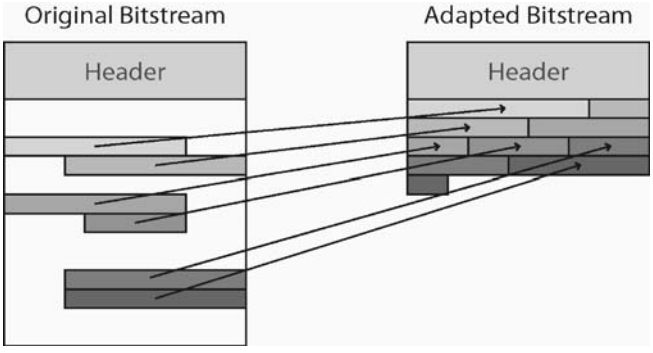

Fig. 3. Adaptation at bit stream level

contextually relevant information. The basis of adaptation within MPEG-21 is quite complex on the outside due to its generic approach and there are several elements that are required in order to perform a complete adaptation (illustrated in Fig. 1). These are as follows:

- **Content digital item (CDI)** – The original encoded format of a CDI is required (although it can be stored in other elements, it is simpler if it is treated separately). This DI should be the highest quality version and contain all elements required. The adapted CDI (ADCI) is the actual content passed from server to client.
- **Generic bitstream description (gBSD)** – The gBSD file, introduced by Amielh et al. [4, 5], as it is commonly known, is essentially the mapping of each of the "important" elements within the encoded file. It can be generated in many ways, but the simplest is during the encoding of the original DI. It is basically an XML representation marking lengths, offsets and other information that is required for adaptation.
- **XML stylesheet (XSLT)** – The XML stylesheet is basically the method for the adaptation. This stylesheet relates the gBSD to the actual values or preferences given by the user or the terminal. It contains processes for the verification of a gBSD file against the required adaptation process, as well as the adaptation process itself. This is generally an execution routine and must be handcrafted for a particular task or set of tasks.
- **Context digital item (XDI)** – The XDI contains the formatted information from the client (and can also contain information from the server) on the target device, the user preferences and is basically the input control parameters for adaptation.

The adaptation process, shown in Fig. 2, is basically passing information through the adaptation engine, where the adapted file (ACDI) is constructed from elements that have been edited (mainly in the header) and elements that have passed a specific criterion, but generally have remained unchanged. For example, and as illustrated in Fig. 3 from a syntactic perspective, if the adaptation process indicates that frames 5 to 20 should be removed, the header should indicate this change (to maintain consistency – so that the decoder does not crash or become unstable). Frames 1 to 4 and 21+ should pass through the adaptation process without change, but frames 5 to 20 will be dropped and not passed onto the adapted file. Whilst the header might also be adapted, in general most of it will be retained as it contains the information outlining the format of the rest of the packet (although this will depend on the encoding format, layers, and wrappers, etc.).

3.3 Representation of content

As the gBSD is used to define the bit-stream layout on a high level it must basically represent a decoder in XML format. This does not mean that it will decode the bit-stream, but the structure of the bit-stream is important and the adaptation methods must be inline with the lowest level defined in the schema. For example, if the adaptation needs to skip frames, and as MPEG codecs are byte aligned (per frame), it is practical in this case to set a marker at the beginning of each frame; this means that

a frame can be dropped without the need to understand the majority of the payload (this is possibly with the exception of updating the frames-skipped section of the header – although this can be avoided). However, as will be seen in the following sections, the schema is based on a much lower level in order to provide more flexibility.

4 Content preparation

4.1 Multi-resolution model generation

For a multi-resolution model, we adopt and extend the concept of clustering representation [14]. In this section, the idea of clustering is illustrated along with its extension to the compact representation of vertex properties and animation parameters.

The premise involves clustering all the data so that a specific complexity can be obtained by simply choosing a set of clusters. From the complex mesh $M_n(V_n, F_n)$ where V_n is a set of vertices and F_n is a set of faces, it is sequentially simplified to $M_{n-1}, \ldots, M_1, M_0$. A multi-resolution model of this simplification sequence has, or at least is able to generate, a set of vertices V and faces M, where union is denoted as '+' and intersection is denoted as '−':

$$V = \sum_{i=0}^{n} V_i,\ M = \sum_{i=0}^{n} M_i\,, \tag{1}$$

V and M can be partitioned into sets of clusters. The first type is a set of vertices and faces that are removed from a mesh of the level i to make a mesh of the level $i-1$, denoted by $C(i)$. The other type is a set of vertices and faces that are newly generated by simplification, denoted by $N(i)$. Hence a level i mesh is as follows:

$$M_i = M_0 + (\sum_{j=1}^{i} C(j) - \sum_{j=1}^{i} N(j))\,. \tag{2}$$

There are many simplification operators, including decimation, region merging, and subdivision [22]; here we used half edge-collapsing operators [25] and quadric error metrics (QEM) [19]. The error metric is slightly modified to adopt the animation parameters. Each vertex has a measurement of levels of animation. For example, a vertex that is close to the joint in body animation or a vertex that has large facial deformation parameters need to be preserved during the simplification process. At one extreme, it is desired to preserve control points of animation as much as possible. This level of deformation is multiplied by the QEM of each vertex, such that vertices with a high deformation parameter are well-preserved through simplification.

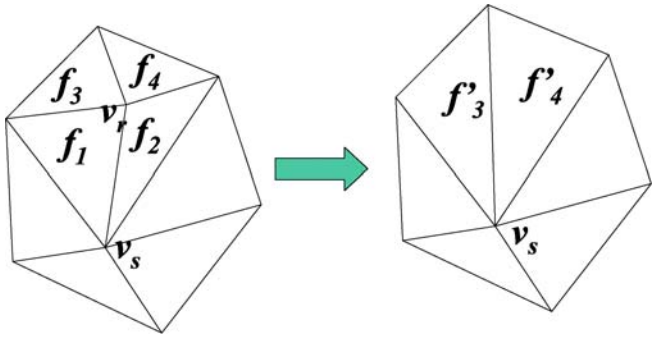

Fig. 4. Illustration of edge-collapsing

By an edge-collapsing operator, an edge (v_r, v_s) is collapsed to the vertex v_s. In the example (Fig. 4), faces f_1, f_2 are removed from the mesh, and faces f_3, f_4 are modified into f_3', f_4'.

The clusters are defined as:

$$C(i) = \{f_1, f_2, f_3, f_4\}$$
$$N(i) = \{f_3', f_4'\}\,.$$

To evaluate Eq. 2 requires setting the union and intersection, which are still complex. Using the properties of the simplification ensures $N(i)$ to be a subset of unions of $M_0, C(1), \ldots, C(i-1)$. Using this property, the cluster $C(i)$ is sub-clustered into a set of $C(i, j)$, which belongs to $N(j)$ where $j > i$ and $C(i, i)$, which does not belong to any $N(j)$. This is same for M_0, where $M_0 = C(0)$. Thus, the level i mesh is represented as Eq. 3, which requires simple set selections.

$$M_i = \sum_{k=0}^{i} (C(k,k) + \sum_{j=i+1}^{n} C(k,j))\,. \tag{3}$$

The last process is the concatenation of clusters into a small number of blocks to reduce the number of selection or removal operations during the adaptation process. Processing vertices is rather straightforward because the edge-collapsing operator (v_i, v_s) ensures that every $C(i)$ has a single vertex v_i as $C(i, i)$. By ordering vertices of $C(i, i)$ by the order of i, the adaptation process of vertex data for level i is a single selection of a continuous block of data, $v_0, v_1, \ldots, v_i$. For the indexed face set, each $C(i)$ is ordered by $C(i, j)$ in the ascending order of j. Thus, an adaptation to level i, consists of at most $3i+1$ selections or at most $2n$ removals of concatenated blocks.

So far, we have described the process using only the vertex positions and face information. In the mesh, there are other properties that have to be taken into account, such as normal, color, and texture coordinates. Because these properties inheritably belong to vertices, a similar process to vertex positions is applied. Exceptional cases

are 1) two or more vertices using the same value for a property and 2) a single vertex having more than two values. In both cases, there is a unique mapping from a vertex and face pair to a value of properties. The cluster $C(i)$ has properties that have mapping from (v_i, f_j), where $v_i \in C(i)$. If a property p belongs to more than one vertex, such as $(v_i, f_1) \rightarrow p$ and $(v_j, f_2) \rightarrow p$, p is assigned to the cluster of $C(j)$, where $j < i$. By ordering this, p remains active as long as there is one vertex that has p as its property. Therefore, we have a valid set of clusters for each level i.

Each cluster has a set of vertices and vertex properties such as vertex normal, colors, and texture coordinates. Along with vertex information, the cluster has a set of indexed faces, normal faces, color faces and texture faces. Also each cluster can consist of sub-segments with their own material and texture. Each level is selected by choosing blocks of clusters.

4.2 Animation data construction

Following the adaptation of the mesh, animation data is applied to the adapted geometry. For the facial animation we use an MPEG-4 compliant facial animation system with facial animation parameters (FAP) driving our facial animation engine. These parameters provide information about the displacement of facial definition parameters (FDP) represented as feature points on the face. Each FAP is used to animate one FDP in one direction. MPEG-4 defines 66 FAP values in order to reproduce a wide range of facial expressions. Each FAP is expressed in terms of facial proportions called facial animation parameters units (FAPU), i.e. the distance between eyes. With this approach, in terms of animation, MPEG-4 is able to provide a set of displacement parameters without relation to a specific face model.

The FAP stream does not provide any information for the displacement of neighboring vertices; therefore, for each FAP, we use a method that defines each displacement, i.e. which vertices are influenced and in which direction according to FAP intensities. MPEG-4 provides a referencing method called face definition tables, and is based on a piece-wise linear interpolation in order to animate the face model. These tables (also referred to as facial animation tables or FAT) provide information about which vertices should be translated or rotated for each FAP displacement. For a translation or rotation a face definition transform node contains information on which part of the scene-graph is under influence. In order to define which vertices are influenced by FAP shape deformation, we use the face definition mesh structure to determine which vertices are animated according to the current FAP. In our method, we use a single interval boundary between consecutive frames, however we use different multiple interval boundaries in order to represent different amplitudes of deformation according to FAP intensity. This deformation information is used during the computation of model deformation in order to produce expressions according to a set of FAP values (see Fig. 5 for further details of the structure of the face definition tables)

The use of face definition tables is optional, and the providing definition information for each FAP could be achieved by a designer defining each influence manually; however areas such as the lips (shown in Fig. 6) make this work almost impossible due to the close proximity of 21 FAP values in a very small region. In addition, this kind of manual work would have to be performed for each model and at each level of detail.

Instead, we use another technique that utilizes geometric deformation algorithms to compute FAP influences. This technique allows an automatic computation of influenced vertices according to FDP only, i.e. the position of each feature control point. With algorithms developed based on this technique, in a few seconds, the facial animation engine is able to animate a face model with only this basic information (FDP). The huge advantage of using this kind of approach, is that for different levels of detail, we can simply recalculate the influence information

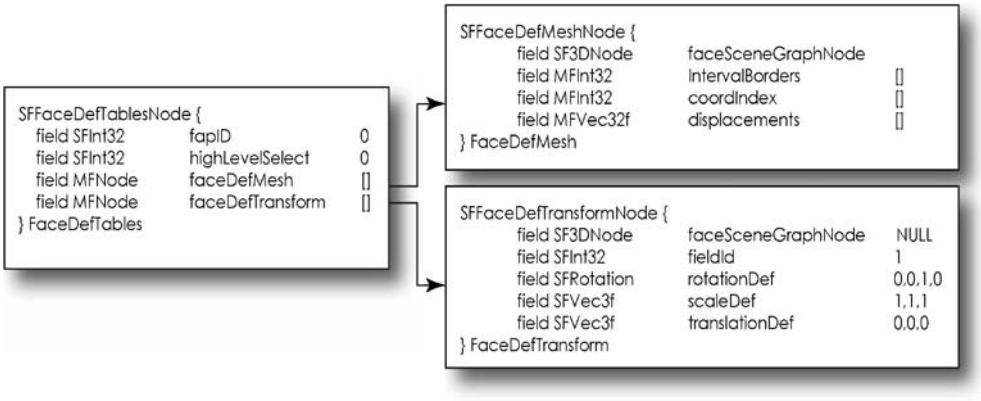

Fig. 5. Example of face definition tables

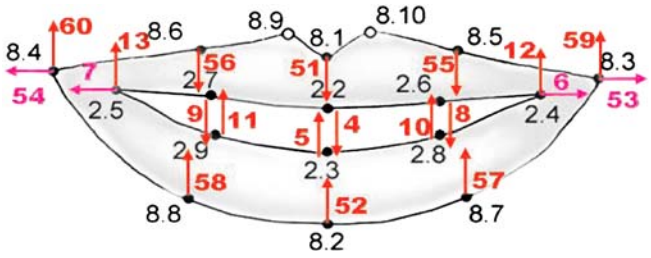

Fig. 6. Animation parameters used in the lip area

by using the same FDP for the same model. Although the influence computation should be done only one time by model and complexity level, due to the computational complexity, this method is not suitable for devices with a low performance processing unit. Instead, the best solution, which can be applied to a wider range of platforms, is to automatically export piece-wise linear interpolation information from a geometric deformation engine and to use it during animation. From the high-level resolution model, we automatically construct the facial animation table, which describes influences of FAP points and information for the interpolation. As the vertices of the model were ordered by FDP and influence, we can easily extract the corresponding FAT information from a high-level FAT table for each model; therefore, only the corresponding part of the global FAT table is transmitted to the client. Figure 7 illustrates the adaptation process for the facial animation.

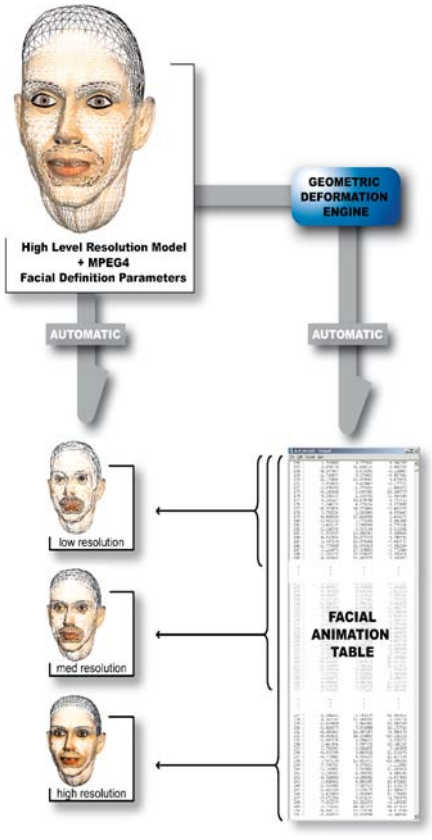

Fig. 7. Adaptation of facial animation

5 MPEG-21 adaptation

5.1 Overview

Whilst MPEG-21 is based on the principle of adaptation from non-appointed peers, general usage is geared towards the concept of a server/client architecture, whereby content is adapted on the server side and streamed to a client.

In this research, not only the geometry but also body and facial animation contents are being adapted for different contexts (Fig. 8). For the adaptation, we focus on the entire context of adaptation instead of a particular context of network capacity or performance of a rendering hardware. As previously described, there are many different target contexts to which the mesh and animation can be adapted; the main consideration is to make sure that they are adapted to satisfy all contexts. As an overview, the network, the device capability, and the user's preferences/restrictions are the main considerations. In the context of the network, we are able to use bandwidth measurements and figures in order to determine the available capacity. The user's preferences generally consist of choosing the LoD explicitly (if the application allows). The terminal's capability is the most complex, and is dependent on many factors; in this case we use a benchmarking device.

5.2 Adaptation through benchmarking

In order to solve one of the key considerations of DIA (i.e. to concisely describe a terminal's computational power), we introduced the concept of benchmarking for both the digital item, and the terminal into the schema. This concept applies not only to graphics, but is generic enough to be used for video and audio media types as well. As part of our initial investigation, we use a linear approximation that is attributed to any digital item, i.e. regardless of the ratio it is applied in a linear fashion with no compensation for extremes or alternative scales. In essence, and in terms of graphics, we use the following equations in order to approximate the adaptation ratio:

$$M_r = F_R / F_D \,. \tag{4}$$

- **Media ratio** $(M_r) \sim$ is the ratio between the maximum value for the media indicator on the reference machine (F_R) and the desired media reference on the user's terminal (F_D). In the case of graphics and video, this will normally be the frame rate (it is in our test case). For example, if F_R is 50 frames per second (fps), and

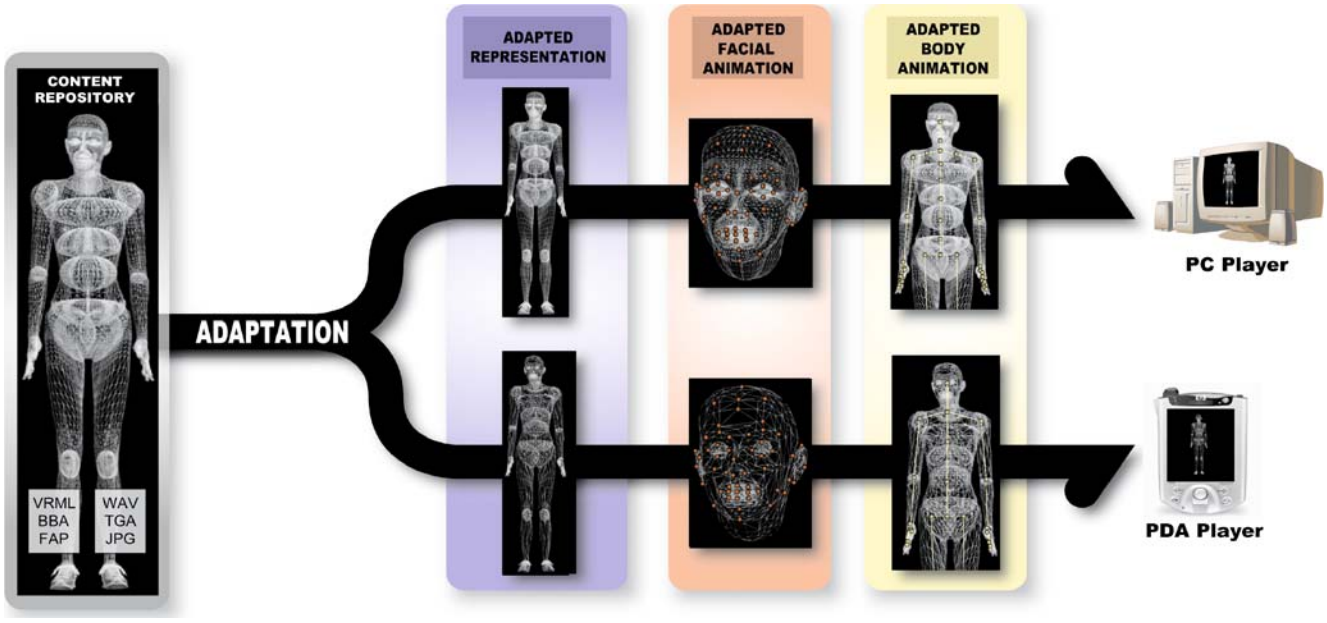

Fig. 8. Adaptation process overview

the desired rate is 25 fps, then the media ratio would be 2.0; assuming the benchmark ratio is above 1.0, no adaptation would be required.

$$D_r = B_R / B_T \,. \tag{5}$$

- **Device benchmark ratio** (D_r) ∼ is the ratio between the maximum benchmark value (B_R) obtained from the reference machine and the same benchmark (B_T) obtained for the user's terminal. Similar conditions should exist for the benchmarking process on the reference machine, and the execution of testing using the media indicator.

$$R_A = M_r \times D_r \times C_R \,. \tag{6}$$

- **Adaptation ratio** (R_A) ∼ provides the linear adaptation value used to adapt the digital content. It consists of the product of the media ratio (M_R), the device benchmark ratio (D_R) and the computation ratio (C_R). C_R is the ratio of computation space given up for processing this media; if the terminal is dedicated to processing this media C_R can be set to 1.0.

The computation ratio is used mainly because the benchmarking process and consequential media indicator (in our case frames per second) on the reference machine are performed using 100% of the CPU (or $C_R = 1.0$) and the user may desire less than 100% usage on their terminal for decoding and playing that specific digital item. Eq. 6 is used for each individual digital item (and in most cases, decoding and rendering are computed separately, C_R is then adjusted accordingly).

Using a conglomeration of Eqs. 4, 5 and 6, we are able to obtain a suitable adaptation ratio that can be used to adapt the media towards a specific device (described in Sect. 3.2). For all cases where R_A is less than 1.0 adaptation is required, for all other cases the mesh and animation remain unchanged.

Our method is very effective because it permits adaptation without the need to (a) comprehend the various factors within the computer (e.g. CPU type, frequency, etc., which places several arbitrary assumptions on the hardware architecture) and (b) reduces the process to a simple linear approximation.

Benchmarking assumes the presence of a capable rendering engine, for example whilst the decoders for MPEG-4 graphics may be present; a 3D rendering engine may not – for example: if the user's terminal is a mobile phone. Therefore, in this case transmoding (converting media from one mode to another) is a possible alternative. This is especially so with 3D graphics where quite a wide range of obvious transmoding options are available (3D to 2D Graphics, 3D to video, etc.). Therefore, as part of our investigation we are currently in the process of researching such alternatives; these are explained in more detail in Sect. 7.

5.3 Benchmarking devices

The benchmarking scheme for device benchmarking is illustrated in Fig. 9. Here, the terminal device communicates with its counterpart in the media server to setup connections. Within this framework, the benchmark is executed only once offline (during the initial set-up) and only

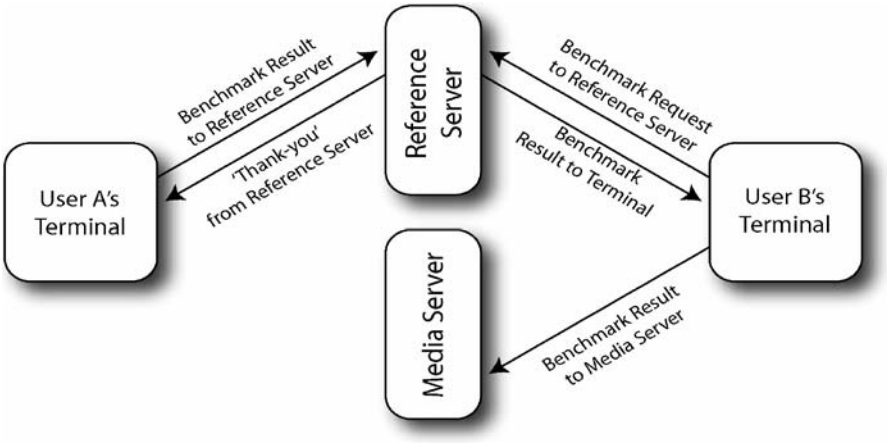

Fig. 9. Demonstration of both user upload of benchmark, and user download of benchmark from reference server

the results are transmitted once the device is connected. It is assumed that every terminal has its own benchmark result when it connects to a network (and ultimately a media server). If the terminal contains no context stating its benchmark, the process system is able to collect one for it, as follows.

In general all benchmarks for a device need to be executed during a suitable period of time; this is usually when the user is not specifically using the device and can expect the benchmark to severely degrade the performance of other processes on their device; i.e. during an installation period. However, as benchmarks consume all the computational power of the device, and as it can be appreciated that the more benchmarks that are executed the better, plus taking into account the understanding of users in general, it is expected that a terminal may connect to a server having possibly only executed a small number of the possible benchmarks. Therefore, the user is unlikely to have downloaded the latest benchmarks as part of the course.

In this case, we assume that at least a base benchmark has been executed during installation; whilst this does not give a clear indication of the performance of the device, it is capable of providing a limited understanding of the user's terminal and is only used as a last resort. In order to tackle the problem of benchmarking requirements, we introduce the concept of a reference server in order to provide preprocessed benchmarks for specific devices. The reference server model is used as follows (and illustrated in Fig. 9).

- **Device benchmarking** $\sim$ a user, who has executed one or more of the standard benchmarks, is asked if they would like to upload these results to the reference server. Upon agreement, the benchmark result and device related information is uploaded to the server.
- **Benchmarking weighting** $\sim$ as with any benchmark result, and due to the possible variables in execution (including device variables, additional processes, and user intervention, etc.), a simple weighting is provided to the benchmark being uploaded in order that a weighted average can be obtained. Currently we are using a simple Bayesian estimate [24]
- **Device benchmark reference** $\sim$ a user, who has not executed a benchmark for their terminal, requests all the relevant benchmarks for their device from the reference server, the reference server returns all available benchmarks to the terminal, which can then be sent to the media server to enable the adaptation to continue. Should a device's benchmark still not be available, the base benchmark will then be used as a last resort.

The use of a reference server is not desirable, however it serves as a function for those user's whose patience for benchmarking, terminal calibration and device testing is limited.

5.4 Adaptation through network capacity

The adaptation of the mesh and animation stream in terms of the network capacity is directly governed by the available bandwidth C_B, the encoded stream size F_S (as the media is generally adapted after it has been encoded), and the download time T_W in seconds that a user is prepared to wait or that a provider believes reasonable; exemplified in Eq. 7.

$$\frac{C_B}{F_s} \times T_w = R_A \,. \tag{7}$$

R_A provides the ratio of the original file to the adapted file, and as the main elements contributing to this size (assuming that the encoding process is well-balanced) is the mesh size, the size of the mesh is reduced. Values for F_S, C_B, and even T_W are easily obtained and, therefore, the value for R_A is also easily determined.

5.5 Additional considerations

For facial animation, after adaptation, the model uses the corresponding facial animation table. During animation, a real-time deformation engine compiles each FAT according to the current set of FAP, and computes deformation of the mesh by merging each FAT. This involves small computational cost for each frame by simplifying the FAT generation process as a simple selection of sub-sets. This technique works well on powerful platforms, but is also appropriate for platforms with a lower processing capability, such as a PDA device.

6 Experimental results

Table 1 shows the benchmark results for different target devices. As a basis for graphics we used the ViewPerf benchmark from SPEC [36] and compared the values with the performance of an actual application, called VHD++ [34] with virtual constructions and character animation. The benchmark result of each ViewPerf test set is averaged using a geometric sum [36].

Fig. 10 clearly illustrates that the averaged benchmark result approximates to the actual performance of the real application (with the exception, although within acceptable limits, of application test # 2, on machine 4). By utilizing different benchmark results for different data sets and applications, we can approximate the performance more closely. For this case, application # 2 could utilize a subset of benchmark such as light-06 and 3dsmax-02, which will describe its performance more closely. In order to verify proposed method, we applied a set of human body models with both body and face animation. Multi-resolution models were generated based on the benchmark for that device and the resulting size values are tabulated in Table 2 and illustrated in Fig. 11.

The progressive mesh approach is known as a method of near optimal storage usage for multi-resolution models. The discrete mesh is a representation of a set of discrete levels of details, which is still quite common in most of real-world applications because of its simplicity in adaptation.

The numbers are the size of the VRML and BIFS files, respectively. Since the PM cannot currently be encoded in the BIFS format, only the approximated size for the VRML file is noted. As a result of the adaptation, the highest details have a number of polygons of 71 K and 7 K for each model, whilst the lowest details have 1K and 552 polygons each (Fig. 12). The models are constructed to have five different levels. The body model has 12 operations of adaptation for each segment, which has vertex normal and texture coordinates as properties.

Table 1. Benchmark value and application performance

	3dsmax-02	drv-09	dx-08	light-06	proe-02	ugs-03	Geometric sum	App. 1	App. 2
1	17.05	68.31	83.48	26.98	15.79	19.75	30.58	37.0	40.0
2	13.41	46.50	59.07	14.47	12.19	17.39	21.99	30.7	33.2
3	7.80	14.93	39.38	12.46	4.29	6.31	10.75	16.20	25.4
4	6.76	31.78	35.52	10.38	7.95	5.70	12.37	15.3	14.9
5	0.36	0.92	1.43	0.75	0.47	0.23	0.58	0.4	0.6
6	4.53	16.42	23.04	6.51	4.47	4.77	7.87	14.2	17.3

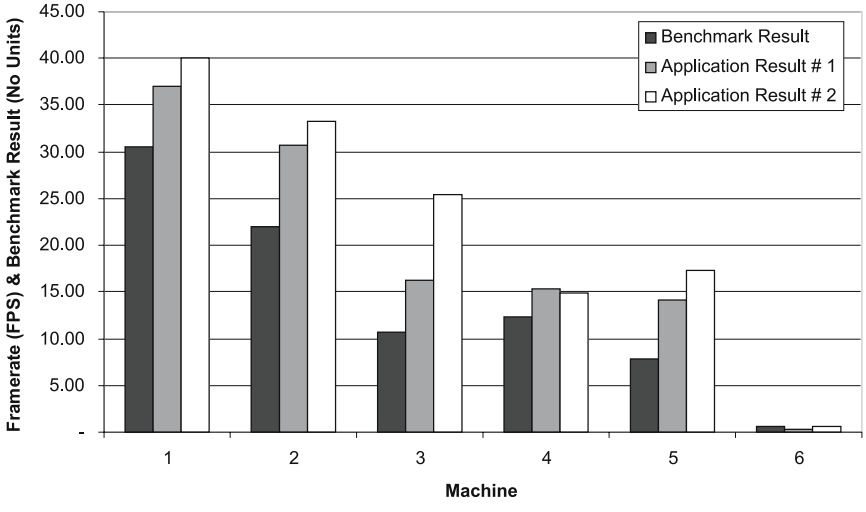

Fig. 10. Benchmark and application performance results

Table 2. Model data size

	No. of polygons	Original model	Proposed method	Progressive mesh	Discrete mesh
Body	71 K	12.7 M / 4.5 M	17.5 M / 6.5 M	∼ 13 M	51.6 M / 12.0 M
Face	7 K	0.8 M / 0.3 M	1.0 M / 0.5 M	∼ 0.9 M	3.9 M / 1.6 M

Fig. 11. Example models

This number of operations is quite small when compared to operations for a PM, which requires at least $n/2$ operations, where n is number of vertices (28 K for the body model). Furthermore, the method uses a simple selection, whilst PM requires relatively complex substitutions. The proposed method is located in-between of these approaches, and is flexible and simple enough to allow adaptation with relatively small file size. Although the proposed method has larger data to the PM approach, it is encodable to standard codec and is able to be transmittable via standard MPEG streams. It also utilizes a simpler adaptation mechanism, which is very similar to the simplest discrete level selection.

7 Current and future research

The context within MPEG-21 digital item adaptation is very generic, and in many respects does not even need to apply to MPEG encoding/representation schemes; therefore our current research is aimed at determining if there are better approximations for the benchmarked result and the polygon mesh adaptation, possibly based on a logarithmic schema. We are also looking at linking the results more closely with other sets of terminal descriptors, such as display capabilities (e.g. defining the screen resolution, etc.), and memory capacity. A straightforward continuation of such an adaptation would be to handle other media types for the lowest levels, i.e. when the device is not capable of 3D rendering, to deliver a media that it can manage. We are investigating automatic conversion of 3D animation to animated 2D vector graphics, for example, illustrated in Fig. 13, since many mobile phones support such formats.

The difficulties in the 3D-to-2D transmoding are to maintain intrinsic 3D properties, such as depth perception and lighting in the 2D representation, and to keep the size of generated data small enough for practical use. We are also concerned with the dynamic adaptation of both representation and animation in harmony with each other. This mainly involves transmitting a representation based on the level of articulation (LoA) used, which greatly reduces the overall polygon count; whilst at the same time maintaining the possibility for a dynamic increase in LoA. Last, we are currently researching methods for

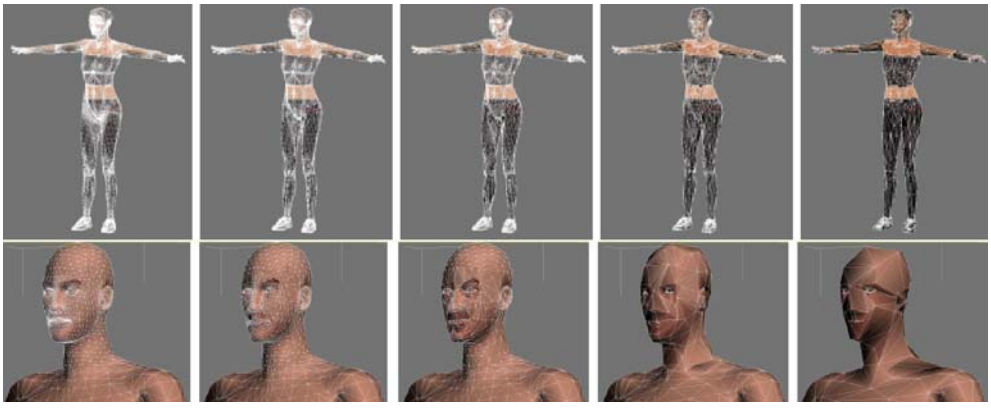

Fig. 12. Examples of adapted meshes (top from left: body models with 71 K, 45 K, 30 K, 15 K and 5 K polygons) (bottom from left: face models with 7 K, 5 K, 3 K, 2 K and 0.5 K polygons)

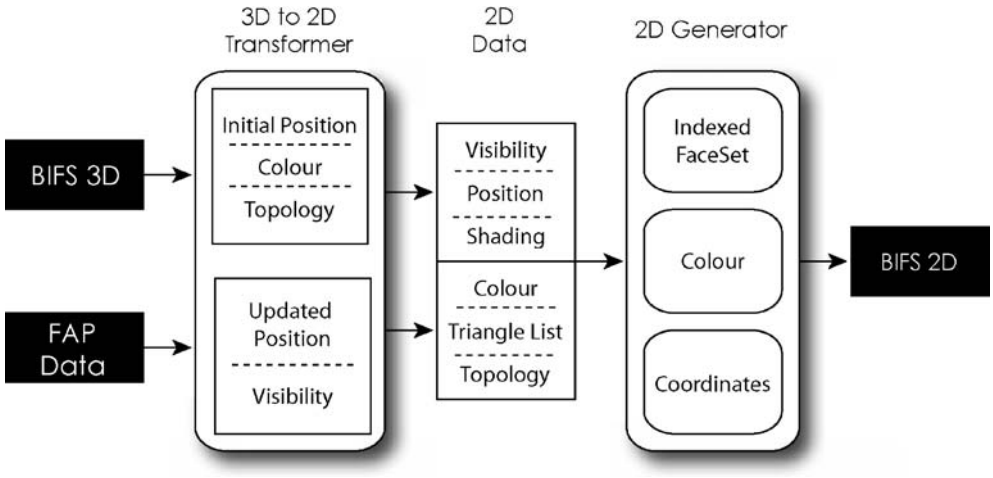

Fig. 13. Overview of a possible 3D-to-2D architecture for MPEG facial animation

adapting multi-resolution textures suited for similar applications.

8 Conclusion

In this paper we have presented a method to format 3D representation data that can then be encoded and adapted using a generic schema within the MPEG-21 framework; essentially the representation is produced in such a way that it requires no additional information on the client's side and can be rendered immediately. In addition, the representation used is suitable for clients who have low computational power and it provides significant advantages over progressive mesh techniques of devices such as PDAs. We have also introduced a benchmarking method that permits a linear approximation to be used for the adaptation process of the mesh representation, reducing the overall polygon density to match the capabilities of the device. Whilst the method presented only approximates the adaptation of a representation's polygon density to a specific value, this approximation is well within acceptable limits and it is not intended, nor could it be hoped, to be more precise.

Acknowledgement The research presented was funded through the European Project DANAE by the Swiss Federal Office for Education and Science (OFES). The authors would also like to thank Lionel Egger for his valuable test models and consultation.

References

1. Adelson, E.: Mechanisms for motion perception. Optics & Photonics News **2**(8), 24–30 (1991)
2. Aggarwal, A., Regunatha, S., Rose, K.: Compander domain approach to scalable AAC. Proceedings 110-th Audio Engineering Society Convention (2001)
3. Ahn, J., Wohn, K.: Motion Level-of-Detail: A simplification method for crowd scene. Proceedings Computer Animation and Social Agents (CASA), pp. 129–137 (2004)
4. Amielh, M., Devillers, S.: Multimedia content adaptation with XML. Proceedings International Conference on Multimedia Modeling (MMM), pp. 127–145 (2001)
5. Amielh, M., Devillers, S.: Bitstream syntax description language: application of xml-schema to multimedia content adaptation. Proceedings 11-th International WWW Conference: CDROM (2002)
6. Berka, R.: Reduction of Computations in physics-based animation using level of detail. Spring Conference on Computer Graphics, pp. 69–76. Comenius University (1997)
7. Carlson, D., Hodgins, J.: Simulation levels of detail for real-time animation. Proc. Graphics Interface, pp. 1–8. ACM (1997)
8. Chen, B., Nishita, T.: Multiresolution streaming mesh with shape preserving and QoS-like controlling. Proceedings 3D Web Technology, pp. 35–42. ACM (2002)
9. Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal P., Brooks, F., Wright, W.: Simplification envelopes. Proceedings ACM SIGGRAPH, pp. 119–128 (1996)
10. Cohen, J., Olano, M., Manocha, D.: Appearance preserving simplification. Proceedings ACM SIGGRAPH, pp. 115–112 (1998)
11. Debunne, G., Desburn, M., Cani, M., Barr, A.: Dynamic real-time deformations using space and time adaptive sampling. Proceedings ACM SIGGRAPH, pp. 31–36 (2001)
12. DeHaemer, M., Zyda, M.: Simplification of objects rendered by polygonal approximations. Comput. Graphics **15**(2), 175–184 (1991)
13. Di Giacomo, T., Capo, S., Faure, F.: An interactive forest. Proceedings Eurographics Workshop on Computer Animation and Simulation, pp. 65–74 (2001)
14. Di Giacomo, T., Joslin, C., Garchery, S., Magnenat-Thalmann, N.: Adaptation of virtual human animation and representation for MPEG. Comput. Graphics **28**(4), 65–74 (2004)
15. Distler, H., Gegenfurtner, K., VanVeen, H., Hawken, M.: Velocity constancy in a virtual reality environment. Perception **29**(12), 1423–1435 (2000)
16. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary

meshes. Proceedings ACM SIGGRAPH, pp. 173–182 (1995)
17. Fogel, E., Cohen-Or, D., Ironi, D., Zvi, T.: A web architecture for progressive delivery of 3d content. Proceedings 3D Web Technology, pp. 35–41. ACM (2001)
18. Funkhouser, T., Sequin, C.: Adaptive display algorithms for interactive frame rates during visualization of complex virtual environments. Proceedings ACM SIGGRAPH, pp. 247–254 (1993)
19. Garland, M., Heckbert, P.: Simplifying surfaces with color and texture using quadric error metrics. Proc. IEEE Visualization, pp. 263–270 (1998)
20. Giang, T., Mooney, R., Peters, C., O'Sullivan, C.: ALOHA: adaptive level of detail for human animation towards a new framework. Proceedings Eurographics, pp. 71–77 (2000)
21. Granieri, J., Crabtree, J., Badler, N.: Production and playback of human figure motion for visual simulation. ACM Trans. on Modeling and Computer Simulation **5**(3), 222–241 (1995)
22. Heckbert, P., Garland, M.: Multiresolution modeling for fast rendering. Proceedings Graphics Interface, pp. 43–50 (1994)
23. Heckbert, P., Rossignac, J., Hoppe, H., Schroeder, W., Soucy, M., Varsney, A.: Multiresolution surface modeling course. ACM SIGGRAPH, Course 25 (1997)
24. Hoeting, J., Madigan, D., Raftery, A., Volinsky, C.: Bayesian model averaging: a tutorial. Statist. Sci. **14**(4), 382–417 (1999)
25. Hoppe, H.: Progressive meshes. Proceedings ACM SIGGRAPH, pp. 99–108 (1996)
26. Hoppe, H.: View-dependent refinement of progressive meshes. Proceedings ACM SIGGRAPH, pp. 189–198 (1997)
27. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. Proceedings ACM SIGGRAPH, pp. 19–26 (1993)
28. Hutchinson, D., Preston, M., Hewitt, T.: Adaptive refinement for mass/spring simulations. Proc. EUROGRAPHICS Workshop on Computer Animation and Simulation, pp. 31–45 (1996)
29. Joslin, C., Magnenat-Thalmann, N.: MPEG-4 animation clustering for networked virtual environments. IEEE Conference on Multimedia and Expo (ICME): CDROM (2004)
30. Kourtzi, Z., Shiffrar, M.: Dynamic representations of human body movement. Perception **28**(1), 49–62 (1999)
31. Lindstrom, P., Turk, G.: Image-driven mesh optimization. ACM Trans. on Graph. **19**(3), 204–241 (2000)
32. Multimedia Framework (MPEG-21) Part 7: Digital Item Adaptation; ISO/IEC JTC 1/SC 29/WG 11/FDIS 21000-7:2004
33. Ohshima, T., Yamamoto, H., Tamura, H.: Gaze-directed adaptive rendering for interacting with virtual space. Proceedings Virtual Reality Annual International Symposium (VRAIS), pp. 103–110. IEEE Computer Society (1996)
34. Ponder, M., Papagiannakis, G., Molet, T., Magnenat-Thalmann, N., Thalmann, D.: VHD++ development framework: towards extendible, component based vr/ar simulation engine featuring advanced virtual character technologies. Proceedings Computer Graphics International (CGI), pp. 96–104. IEEE Computer Society (2003)
35. Soucy, M., Laurendeau, D.: Multiresolution surface modeling based on hierarchical triangulation. Comput. Vision Image Understand. **63**(1), 1–14 (1996)
36. SPECViewPerf 7.1.1: http://www.spec.org/gpc/opc.static/viewperf71info.html





PROF. HYUNGSEOK KIM was a senior researcher at MIRALab, University of Geneva. He received his PhD in Computer Science in February 2003 at VRLab, KAIST. He is currently an Assistant Professor at Department of Internet & Multimedia Engineering, Konkuk University, Korea. His main research field is real-time interaction in virtual environments, more specifically multi-resolution modeling of shape and texture and multi-modal interaction mechanisms. He has actively participated in several European Projects focused on topics of shape modeling, multi-modal interaction and evoking believable experiences in the virtual environment.

PROF. CHRIS JOSLIN is currently an Assistant Professor at the School of Information Technology, at Carleton University, Canada. He holds Master's degrees in Engineering (University of Bath, United Kingdom), and Computer Science (University of Geneva, Switzerland). He obtained his PhD in Information Systems, under the supervision of Professor Nadia Magnenat-Thalmann, also from the University of Geneva. He is the author of several journal papers, book chapters, and conference papers in subjects ranging from collaborative virtual environments, spatial audio models, to media adaptation. He is currently a member of the Standards Council Canada (SCC), representing the Canadian National Body in the standardisation of Still (SC29WG1) and Dynamic (SC29WG11, more commonly known as MPEG) media, mainly focusing on research in the area of MPEG-4 (media compression) and MPEG-21 (media control). His current research involves immersive collaborative environments (unrestricted collaborative spaces, virtual reality devices, and visualisation techniques), dynamic media adaptation (dynamic sessions, scalable media, dynamic contexts, and session mobility issues), and interactive media (in-car interfaces, context-based emergency services information displays, and head-mounted displays).

THOMAS DI GIACOMO obtained his Master's thesis in September 2001 at the University of Grenoble (iMAGIS, INRIA/INPG/CNRS) on Computer Graphics. Since March 2002, he has been working as a research assistant and PhD candidate at MIRALab. His main interests involve cloth animation, realtime and level of details for animation, and physically-based animation.

DR. STEPHANE GARCHERY is a computer scientist who studied at the University of Grenoble and Lyon, France. He is working at the University of Geneva as a senior research assistant in MIRALab, participating in research on facial animation for real time applications. One of his main tasks is focused on developing an MPEG-4 facial animation engine, applications and tools to automatic facial data construction. He has developed different kinds of facial animation engines based on MPEG-4 facial animation parameters for different platforms (stand-alone, web applet and mobile devices), and different tools for fast design in an interactive way.

PROF. NADIA MAGNENAT-THALMANN has pioneered research into virtual humans over the last 25 years. She obtained several Bachelor's and Master's degrees in various disciplines (psychology, biology and chemistry) and a PhD in Quantum Physics from the University of Geneva. From 1977 to 1989 she was a Professor at the University of Montreal and led the research lab MIRALab in Canada. She moved to the University of Geneva in 1989, where she founded the Swiss MIRALab, an internationally interdisciplinary lab composed of about 30 researchers. She is author and coauthor of a very large number of research papers and books in the field of modeling virtual humans, interaction with them and in augmented life. She has received several scientific and artistic awards for her work, mainly on the Virtual Marylin and the film Rendez-vous a Montreal. More recently, in 1997, she was elected to the Swiss Academy of Technical Sciences, and was nominated as a Swiss personality who has contributed to the advance of science in the 150 years history on the CD-ROM produced by the Swiss Confederation Parliament.